



US Patent & Trademark Office

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

recursive parser

SEARCH


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)
Terms used **recursive parser**Found **14,032** of **141,345**

Sort results by

relevance

Display results

expanded form

☒ Save results to a Binder

☒ Search Tips

☐ Open results in a new window
Try an [Advanced Search](#)Try this search in [The ACM Guide](#)

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐

1 [rdp—an iterator-based recursive descent parser generator with tree promotion operators](#)

Adrian Johnstone, Elizabeth Scott

September 1998 **ACM SIGPLAN Notices**, Volume 33 Issue 9Full text available: pdf(602.74 KB) Additional Information: [full citation](#), [abstract](#), [index terms](#)

rdp is a parser generator which accepts *Iterator Backus Naur Form* productions decorated with attributes and ANSI-C actions and produces recursive descent parsers. It has special support for the generation of tree-based intermediate forms, built-in symbol table handling for the implementation of context-sensitive components of the language syntax and a support library that includes a generalised graph handling module that can output graphs in a form suitable for use with well known visuali ...

Keywords: EBNF, LL(1) grammar, derivation tree, iterator, parser generator, tree promotion operator

2 [An object oriented approach to constructing recursive descent parsers](#)

Matthew S. Davis

February 2000 **ACM SIGPLAN Notices**, Volume 35 Issue 2Full text available: pdf(490.03 KB) Additional Information: [full citation](#), [abstract](#), [index terms](#)

We discuss a technique to construct a recursive descent parser for a context free language using concepts found in object oriented design and implementation. A motivation for the technique is given. The technique is then introduced with snippets of a Smalltalk implementation. Some advantages and disadvantages of the technique are examined. Finally some areas of possible future work are discussed.

Keywords: Greibach normal form, context free grammar, design patterns, object oriented, recursive descent parser, smalltalk

3 [Non-deterministic recursive ascent parsing](#)

René Leermakers

April 1991 **Proceedings of the fifth conference on European chapter of the Association for Computational Linguistics**Full text available: pdf(549.14 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

BEST AVAILABLE COPY



A purely functional implementation of LR-parsers is given, together with a simple correctness proof. It is presented as a generalization of the recursive descent parser. For non-LR grammars the time-complexity of our parser is cubic if the functions that constitute the parser are implemented as memo-functions, i.e. functions that memorize the results of previous invocations. Memo-functions also facilitate a simple way to construct a very compact representation of the parse forest. For LR(0) gram ...

4 Recursive-ascent parsing

Larry Morell, David Middleton

June 2003 **Journal of Computing Sciences in Colleges**, Volume 18 Issue 6

Full text available: pdf(72.50 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

A new procedural method of implementing bottom-up parsing called *recursive-ascent parsing* is defined. In recursive-ascent, procedures are used to implement bottom-up parsing in a manner similar to the way procedures are used in recursive-descent to implement top-down parsing. We hypothesize that this makes bottom-up parsing techniques more readily accessible to students. The paper illustrates recursive-ascent techniques and shows how to develop these as a generalization of recursive-desce ...

5 Global Context Recovery: A New Strategy for Syntactic Error Recovery by Table-Drive Parsers

Ajit B. Pai, Richard B. Kieburtz

January 1980 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 2 Issue 1

Full text available: pdf(1.59 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Described is a method for syntactic error recovery that is compatible with deterministic parsing methods and that is able to recover from many errors more quickly than do other schemes because it performs global context recovery. The method relies on fiducial symbols, which are typically reserved key words of a language, to provide mileposts for error recovery. The method has been applied to LL(1) parsers, for which a detailed algorithm is given, and informally proved correct. The algorithm ...

6 Generation of LR parsers by partial evaluation

Michael Sperber, Peter Thiemann

March 2000 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 22 Issue 2

Full text available: pdf(411.54 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#), [review](#)

The combination of modern programming languages and partial evaluation yields new approaches to old problems. In particular, the combination of functional programming and partial evaluation can turn a general parser into a parser generator. We use an inherently functional approach to implement general LR(k) parsers and specialize them with respect to the input grammars using offline partial evaluation. The functional specification of LR parsing yields a concise implementat ...

Keywords: LR parsing, continuations, functional programming, parser generation, partial evaluation

7 Are LR parsers too powerful?

P Machanick

June 1986 **ACM SIGPLAN Notices**, Volume 21 Issue 6

Full text available:  [pdf\(382.91 KB\)](#) Additional Information: [full citation](#), [abstract](#), [index terms](#)

The general trend in the development of parser theory is in the direction of exploring implementing methods of increasing power. In particular, ways of improving the efficiency of LR parsers and the generation of LR tables have been receiving a lot of attention. The value of increasingly powerful tools is questioned from the point-of-view of the need to keep definitions of languages understandable to the programmer. Consideration is given to Wirth's contention that recursive descent is the metho ...

8 [A parser that doesn't](#)

S. G. Pulman

March 1985 **Proceedings of the second conference on European chapter of the Association for Computational Linguistics**

Full text available:  [pdf\(683.26 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citing](#)s

 [Publisher Site](#)

This paper describes an implemented parser-interpreter which is intended as an abstract formal model of part of the process of sentence comprehension. It is illustrated here for Phrase Structure Grammars with a translation into a familiar type of logical form, although the general principles are intended to apply to any grammatical theory sharing certain basic assumptions, which are discussed in the paper. The procedure allows for incremental semantic interpretation as a sentence is parsed, and ...

9 [Parsers in ML](#)

Michel Mauny, Daniel de Rauglaudre

January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM conference on LISP and functional programming**, Volume V Issue 1

Full text available:  [pdf\(940.08 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

We present the operational semantics of streams and stream matching as discussed in. Streams are data structures such as lists, but with different primitive operations. Streams not only provide an interface to usual input/output channels, but may used as a data structure per se, holding any kind of element. A special pattern matching construct is dedicated to streams and the actual matching process will be called parsing. The primary parsing s ...

10 [SNACC: a parser generator for use with Miranda](#)

D. A. Turner

February 1996 **Proceedings of the 1996 ACM symposium on Applied Computing**

Full text available:  [pdf\(725.15 KB\)](#)

Additional Information: [full citation](#), [references](#), [index terms](#)

Keywords: Miranda, attribute grammars, parser generator

11 [From recursive ascent to recursive descent: via compiler optimizations](#)

George H. Roberts

April 1990 **ACM SIGPLAN Notices**, Volume 25 Issue 4

Full text available:  [pdf\(221.05 KB\)](#)

Additional Information: [full citation](#), [references](#)

12 [Recursive ascent: an LR analog to recursive descent](#)

G. H. Roberts

August 1988 **ACM SIGPLAN Notices**, Volume 23 Issue 8

Full text available:  [pdf\(341.88 KB\)](#) Additional Information: [full citation](#), [citations](#), [index terms](#)

13 Comparative efficiency of general and residual parsers

Frank G. Pagan

April 1990 **ACM SIGPLAN Notices**, Volume 25 Issue 4


Full text available:  [pdf\(546.43 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#)

Some fundamentals of the partial computation concept are concisely reviewed, including its relevance to the relationship between table-driven, general syntactic analyzers and source-language-specific, residual syntactic analyzers. A manual methodology for converting general parsers into generators of residual parsers is explained, using an LL(1) parser as a detailed example. The results of several experiments are reported, comparing the time and space efficiencies of different general parsers wi ...

14 Incremental generation of parsers

J. Heering, P. Klint, J. Rekers

June 1989 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation**, Volume 24 Issue 7

Full text available:  [pdf\(1.53 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

An LR-based parser generator for arbitrary context-free grammars is described, which generates parsers by need and processes grammar modifications by updating already existing parsers. We motivate the need for these techniques in the context of interactive language definition environments, present all required algorithms, and give measurements comparing their performance with that of conventional techniques.

15 Packrat parsing:: simple, powerful, lazy, linear time, functional pearl

Bryan Ford

September 2002 **ACM SIGPLAN Notices , Proceedings of the seventh ACM SIGPLAN international conference on Functional programming**, Volume 37 Issue 9

Full text available:  [pdf\(171.57 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


Packrat parsing is a novel technique for implementing parsers in a lazy functional programming language. A packrat parser provides the power and flexibility of top-down parsing with backtracking and unlimited lookahead, but nevertheless guarantees linear parse time. Any language defined by an LL(k) or LR(k) grammar can be recognized by a packrat parser, in addition to many languages that conventional linear-time algorithms do not support. This additional power simplifies the handling ...

Keywords: Haskell, backtracking, lexical analysis, memoization, parser combinators, scannerless parsing, top-down parsing

16 On the covering of left recursive grammars

A. Nijholt

January 1977 **Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages**

Full text available:  [pdf\(767.24 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#)

In this paper we show that some prevailing ideas on the elimination of left recursion in a context-free grammar are not valid. An algorithm and a proof are given to show that every proper context-free grammar is covered by a non-left-recursive grammar.


BEST AVAILABLE COPY

Keywords: context-free, cover, grammar, left-recursion, parsing

17 Parsing and compiling using Prolog

Jacques Cohen, Timothy J. Hickey

March 1987 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,
Volume 9 Issue 2

Full text available:  pdf(2.83 MB)


Additional Information: [full citation](#), [abstract](#), [references](#), [citings](#), [index terms](#), [review](#)

This paper presents the material needed for exposing the reader to the advantages of using Prolog as a language for describing succinctly most of the algorithms needed in prototyping and implementing compilers or producing tools that facilitate this task. The available published material on the subject describes one particular approach in implementing compilers using Prolog. It consists of coupling actions to recursive descent parsers to produce syntax-trees which are subsequently utilized ...

18 The Construction of Stack-Controlling LR Parsers for Regular Right Part Grammars

Wilf R. LaLonde

April 1981 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,
Volume 3 Issue 2

Full text available:  pdf(2.22 MB)

Additional Information: [full citation](#), [references](#), [index terms](#)

19 Experimental comparison of some parsing methods

Robert Gerardy

August 1987 **ACM SIGPLAN Notices**, Volume 22 Issue 8

Full text available:  pdf(725.36 KB)

Additional Information: [full citation](#), [index terms](#)

20 Control structure aptness: A case study using top-down parsing

Gary Lindstrom

May 1978 **Proceedings of the 3rd international conference on Software engineering**

Full text available:  pdf(739.28 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citings](#), [index terms](#)

The range of control structures available in a higher-level programming language directly governs the set of algorithms conveniently programmable therein. This fact has been well-demonstrated by the salutary effect the ideas of structured programming have had on traditional control structures (sequential, iterative, and procedural). This paper seeks to demonstrate this same fact for more advanced control structures through the use of top-down parsing as a case study. A series of increasingl ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)

REST AVAILABLE COPY